

VIRUS BULLETIN

THE INTERNATIONAL PUBLICATION ON COMPUTER VIRUS PREVENTION, RECOGNITION AND REMOVAL

Editor: **Nick FitzGerald**

Editorial Assistant: **Francesca Thorne**

Technical Editor: **Jakub Kaminski**

Consulting Editors:

Ian Whalley, Sophos Plc, UK

Richard Ford, IBM, USA

Edward Wilding, Network Security, UK

IN THIS ISSUE:

• **Speaka-da-lingo:** Following the spate of complex multipartites featured in recent virus analyses, this month we received news of Esperanto. This shows signs of a move to cross-platform capabilities. See p.3.

• **See a good scanner?** The standalone product review focuses on the new version of *Inoculan* – the first major upgrade since its procurement by *Computer Associates*. Our reviewer's findings start on p.21.

CONTENTS

EDITORIAL

Fragmentary Evidence 2

VIRUS PREVALENCE TABLE 3

NEWS

1. CAT-and-mouse 3

2. Enter Esperanto 3

3. Errata 3

IBM PC VIRUSES (UPDATE) 4

VIRUS ANALYSES

1. Zohra the Geek? 6

2. What a Mess! 8

3. Fighting Talk 10

ADDENDUM

Memorial Revisited 9

PRODUCT REVIEW

Inoculan AntiVirus v5.0 for Windows 95 13

END NOTES AND NEWS 16

EDITORIAL

Fragmentary Evidence

Several events during the last month or so set me pondering: What is the correct way to deal with files *affected* by a virus, but not *infected*? The two previous editors have also discussed this.

It may seem unusual to the casual observer, but it is not uncommon for viruses to fail to infect their targets properly. Some file infectors incorrectly calculate the necessary offsets when modifying the headers or entry points of their hosts, which usually results in a corrupted file. At least, corrupted from the victim's point of view – attempts to run such 'infected' programs typically cause your operating system to hang or to trap some kind of exception.

“*Astute anti-virus software users have more than one scanner at their disposal...*”

My predecessors agreed that whilst these modified files are not technically viruses, it is desirable that they should be 'detected'. As a user of anti-virus software, you would at least want these files identified when cleaning up after a virus outbreak. You will have your backups handy or be investigating just what to re-install (although the cautious would advocate 'do the lot regardless').

Unfortunately, there does not seem to be a great deal of agreement among the developers about this. Some take the highly purist approach, only identifying 'infective' examples. Some distinguish two classes of 'infection' – viruses and files modified by viruses. Others try to identify all files affected by a given virus, but do not distinguish between the infective and non-infective resultants when reporting. Those responsible for making these decisions defend them eloquently.

I can see all sides and have some sympathies with each position, but the purist in me inclines to the 'find both and identify them differently' position. Within the industry this seems to be treated as a purely technical issue, but there is another side to it. Astute anti-virus software users have more than one scanner at their disposal (the infamous JammyScan and SkammyScan). On finding 'Virus X' with SkammyScan, they will break out JammyScan. If the latter says 'damaged by Virus X' they will likely not be too confused (a quick look at the fine on-line help will explain).

However, if JammyScan says 'clean' they are in a quandary. Maybe JammyScan does not detect Virus X yet? Let's check its virus list. Assuming the naming problem does not apply and that JammyScan does detect Virus X (which the users cannot easily tell), they now have to decide whether SkammyScan has a false positive, JammyScan a false negative, or SkammyScan has imprecise reporting and may have found a non-infective but virus-related change to the file and JammyScan is highly purist, only reporting viable infections? Not the most helpful outcome from tools that should make our computing experience a smoother, safer ride!

'And not that common?', I'd guess you are thinking. I agree, but suspect it is becoming more so with the rise in distribution of macro viruses. Some early *Word* macro virus disinfectors simply flipped the template bit of infected documents off, leaving the virus macros intact but 'adrift' (*Word* 6/7 only loads macros from templates). Some vendors, not having mastered the structure of *Word*'s OLE2 files, used brute-force scanning, rather than just looking at the parts that could contain macros. These scanners are unable to disinfect 'properly' and some very ugly tricks have been used. OLE2 files are essentially self-contained file-systems, and like their disk-based equivalents, have slack space that can contain 'garbage' from the unfilled part of a memory buffer. Scraps of macro code can thus appear in places *Word* will never see, but brute-force scanners will.

Many other problems have arisen, meaning a lot of documents exist with 'fragments' or 'remnants' of some virus. As a result, some programs scan all *Word* documents, while others only scan those *Word* 6/7 documents that are templates (*Word* 8 documents can contain active macros without being templates). Some people still use brute-force scanners. So, with macro viruses we have the macro-specific complexities crossed with those created by one scanner saying Virus X, another saying 'remnant of Virus X', and another saying nothing because it only reports things that replicate.

Fertile ground for confusion, I'd say!

NEWS

CAT-and-mouse

'As smaller is quickly becoming better, hand-held PCs are not only cute and compact, but are serious, efficient, and valuable...' but Israel-based *Iris Software* stresses the vulnerability of these popular, small systems to malicious programs and virus threats. On 1 December *Iris* released the beta version of *Iris AntiVirus for Windows CE*, incorporating its Compact AntiVirus Technology, or CAT. Currently shipped by the majority of hand-held computer manufacturers including *Compaq*, *Casio*, *Hewlett-Packard*, *NEC* and *Hitachi*, *Microsoft Windows CE* is seen by *Iris* as fast becoming an industry standard.

Self-professed pioneers of the first ever anti-virus solution back in 1987, *Iris* claims first place again with the launch of this new product. Full program details are available on the company's Web site at <http://www.irisav.com/> ■

Enter Esperanto

A virus with the potential to infect most popular desktop computers was discovered in late November. *Central Command Inc*, US distributors of *AntiViral Toolkit Pro* (AVP) has circulated information concerning the new virus, named Esperanto.4733. *Command's* president, Keith Peer, claimed it is the first of its kind, although labelling it cross-platform may be a little premature.

In addition to infecting DOS COM and EXE programs, the virus looks for MDEF Macintosh resources and operates under the Macintosh environment. The initial claim that it also infects *Windows* EXE files is incorrect – AVP's head developer confirms that bugs in the virus cause it to corrupt *Windows* hosts when it tries to infect them. It cannot spread from Macintosh to PC, or from PC to Macintosh. When an infected Macintosh program is run, the virus simply ignores the DOS/*Windows* instructions, and *vice versa* ■

Errata

We apologize for misleading information in the November review of *IBM AntiVirus for NetWare*. This concerns the comment that it took *IBMAV* longer to scan the full *VB* virus collection than other products tested recently. As *VB* normally only compares scan times on the Clean test-set, this comment should not have been published. The rationale behind this is simple – scanners very seldom run into a large number of infected files in real-world usage, so reporting comparative performance on the Clean test-set gives the best real-world performance indication.

Please also note that the phrase 'The overall performance... is much lower' (p.19) should have read 'The overhead... is much lower' ■

Prevalence Table – October 1997

Virus	Type	Incidents	Reports
CAP.A	Macro	145	20.8%
Concept	Macro	46	6.6%
Form	Boot	31	4.4%
Laroux	Macro	30	4.3%
NPad	Macro	30	4.3%
AntiEXE	Boot	28	4.0%
AntiCMOS	Boot	27	3.9%
Appder.A	Macro	21	3.0%
Parity_boot.b	Boot	21	3.0%
Dodgy	Boot	19	2.7%
NYB	Boot	19	2.7%
ShowOff	Macro	19	2.7%
Wazzu.A	Macro	16	2.3%
J unkie	Multi	14	2.0%
Empire Monkey	Boot	12	1.7%
Imposter.A	Macro	11	1.6%
Ripper	Boot	11	1.6%
Stoned.Angelina	Boot	11	1.6%
Sampo	Boot	10	1.4%
Parity_Boot	Boot	9	1.3%
Temple.A	Macro	8	1.1%
ExeBug	Multi	7	1.0%
Helper.A	Macro	7	1.0%
MDMA.A	Macro	6	0.9%
Monkey 2	Boot	6	0.9%
Baboon	Boot	5	0.7%
Dzt.A	Macro	5	0.7%
Feint	Boot	5	0.7%
Maverick.2048	File	5	0.7%
Demon.A	Macro	4	0.6%
One-Half.3544g	Multi	4	0.6%
Others		105	15.1%
Total		697	100%

^[1]The Prevalence Table includes three reports each of: Johnny, Kompu, LBB_Stealth, Manzoni, Muck, Natas, Stealth_Boot and Switcher; two reports of each of: ABCD, Bandung, Divina, Edwin, Elvira.239, Goldfish, Int12, Level3.5637, Maverick.1536, Michael-angelo, Moloch, NiceDay, OneHalf, PS-MPC3.603, Rapi, Skim.1455, Spanska.4250, Turbo and Unashamed; and one report of each of: Ant, Assistant, Barrotes, Beryllium, Bleah, Boom, Burglar, Cascade.1701, Doggie, Foetus.1561, Galicia.800, GoldBug, Hiac, HLLP.5850, Hybrid, Int40, Jakarta.559, Jumper.B, Kampana, Leprosy, Maniak, Mess, Nail, Olivia, Outlaw, Pesan, Quandary, Russian_Flag, Sack, Setmd, Simple, Spanska.1000, Stoned.Turbo, Stoned.Lemon, Stoned.O, Stoned.Spirit, Swlabs, Tentacle II, Tentacle.1966, Tequila.2468, TPVO.3783, Urkel and Yankee_Doodle.

IBM PC VIRUSES (UPDATE)

The following is a list of updates and amendments to the *Virus Bulletin Table of Known IBM PC Viruses* as of 15 November 1997. Each entry consists of the virus name, its aliases (if any) and the virus type. This is followed by a short description (if available) and a 24-byte hexadecimal search pattern to detect the presence of the virus with a disk utility or a dedicated scanner which contains a user-updatable pattern library.

Type Codes

C Infects COM files	M Infects Master Boot Sector (Track 0, Head 0, Sector 1)
D Infects DOS Boot Sector (logical sector 0 on disk)	N Not memory-resident
E Infects EXE files	P Companion virus
L Link virus	R Memory-resident after infection

A2S.1160	CR: An appending, 1160-byte virus containing the texts 'C:\DOS' and 'C:\COMMAND.COM'. Infected files end with the string 'ASS'. A2S.1160 CD21 5880 C4A6 CD21 3DCA CA74 74BB 0112 BA20 2303 DA93 CD21
Ale.2354	CEN: An appending, 2354-byte direct infector containing the texts 'Alevirus 97 !!!!!!!!!!! Call Now ???-???? many files from virii service', 'Sao Caetano do Sul', 'Brasil!', 'Ligue Para esta Puta: Viviane' and 'Aberto das 0:00 ate 6:00 A.M'. The payload, which triggers on 19 May 1997, displays the texts with the message 'ALEVIRUS CORINGA' (in text-mode graphics) accompanied by sound effects. Ale.2354 B440 B92F 0990 8D96 0601 CD21 32C0 E827 008D 9631 0ACD 215A
AussieBoy.147	CN: An overwriting, 147-byte direct infector containing the texts '..."AussieBoy" virus from DownUnder...' and '...Coeo ergo sum...'. AussieBoy.147 45E2 F65B B440 B193 BA00 01CD 21B4 3ECD 21B4 4FCD 2173 D3C3
Bashar.670	CR: An encrypted, appending, 670-byte virus, containing the text '[Bashar_Teg] by C.W. - 1997 (JAofM)'. Infected files have the byte 57h ('W') at offset 0003h. While infecting a new file, the virus shows a purple block cursor in the upper right part of the screen. Bashar.670 B922 01BE 1305 8BFE 81C7 6302 9BDB E39B 2EDD 059B 2EDD 159B
Bashar.671	CR: A 671-byte variant. Due to a bug, the virus infects only files with the byte 7Fh at offset 000Fh, and can reinfect hosts. Bashar.671 B923 01BE 3B1E 8BFE 81C7 6402 9BDB E39B 2EDD 059B 2EDD 159B
Boing.349	CN: An encrypted, 349-byte, direct infector containing the texts '*.COM', '[BOING]' and 'TREE'. Boing.349 B9A5 00FA B8?? ??94 5A81 F2?? ??52 5?E2 F7??
Dewbug.938	CR: A stealth, encrypted, appending, 938-byte virus containing the texts 'Give Yosha cold Mountain Dew!', '[Dew-Bug] (C) 1996 Yosha/DC' and 'ANTI-VIR.DAT'. Dewbug.938 BE?? ??B9 8F03 B42F CD21 B802 00F7 E301 C680 349D 46E2 FA??
Eddie.565	CR: An appending, 565-byte virus containing the text 'Dark Avenger, (c) 1997 Eddie'. Eddie.565 B935 02B4 7980 F439 E87C FEB8 0042 33C9 99E8 73FE B479 80F4
Esime.379	CR: An appending, 379-byte virus containing the text 'ESIME VIRUS BY(C). ARV'. The payload, which triggers on 5 March, tries to overwrite 256 sectors of the current disk. Esime.379 B2E9 8816 7F01 B97B 01BA 0000 B440 CD21 B800 4233 D233 C9CD
Gawenda.419	CN: An appending, 419-byte, direct infector containing the texts 'Virus Gawenda (:, 'c:\command.com *.com' and '????????COM'. Infected files start with the byte 90h (NOP). Gawenda.419 B440 B9A3 018D 9409 01CD 21B4 3ECD 21B8 0143 8B8C 6F02 8D94
Harpy.1790	CER: An encrypted, appending, 1790-byte virus containing the texts 'MSDOS' and 'This is a Harpy Viruse..'. Infected files have their time-stamps set to 62 seconds. Harpy.1790 EB1C 83EE 058B FE83 EF10 B970 062E FF74 032E 8A44 032E 3005
Indo.1093	CER: A stealth, 1093-byte appender containing the texts 'SCAN.DAT', 'AVP.OVL', 'CHKLIST.CPS', 'AVP.SET', 'DRV', 'ANTI-VIR.DAT', 'CHKLIST.MS', 'FINDVIRU.DRV', 'SIGN.DEF' and '= Victor Widjaja Virus, (c) VirusIndo ='. Infected files have their time-stamps set to 58 seconds. Indo.1093 1EB4 0BBB 4C53 B94D 41CD 213D 4B4F 7453 06B8 2135 CD21 2E89
Kusumah	CR: Two variants of a stealth, appending virus containing the texts 'KuSuMaH'S JUnJaNi, Bandung', 'C:\COMMAND.COM' and 'CHKDSK'. Infected files have their time-stamps set to 60 seconds for the 3968-byte variant, and to 60 or 62 seconds for the 4269-byte variant. Kusumah.3968 AC03 C481 FE17 0774 F430 0547 E2F2 8ED3 8BE2 1F5F 5E5A 595B Kusumah.4269 B440 B970 0F80 3E12 0101 7503 B99D 0FBA 0001 E8AD 049C E850

- Lilo.1573** **CER:** An appending, 1573-byte virus containing the encrypted texts 'Li-LO.1573 virus v.0 (test) by P&C', 'Divide Error', 'Program too big to fit in memory' and 'COMMAND.COM.EXE'. Infected files have the word 1234h at offset 0003h (COM) or at offset 0012h (EXE).
Lilo.1573 8BFE B961 00FC 51AC B104 D2C0 AA59 E2F6 595F 07C3 33DB B803
- Madjid.2930** **CEMR:** A multi-partite, 2930-byte appender containing the texts '.COM', '.EXE', 'SCAN', 'CLEAN', 'NOD', 'FINDVIRU', 'GUARD', 'VSAFE', 'MSAV', 'CHKDSK', 'G000BE3', 'HAREEB' 'NC', 'OHHHHH... MADJID .', 'I am here .They kill the love .I am solitary.', and 'Here is very dark.HELP ME... HELP ME... HELP...'. The original MBR and the rest of the virus' code is stored in the last seven sectors of track zero. The first pattern is for detection in files and memory; the second for MBRs.
Madjid.2930 B85E 5ECD 213D E5E5 7462 B813 35CD 2189 9CA4 018C 84A6 01B4
Madjid.2930 12B1 06D3 E0BA 8000 8EC0 B9?? 00B8 0602 06CD 13B8 4A02 50CB
- Marawi.2719** **CER:** An appending, 2719-byte virus containing the encrypted texts 'Ang VIRUS na ito ay taosDpuso naming inaalay kay Professor HERMILITO GO ng MSUDIIT. Kung hindi dahil sa kanyang KAHAMBUGAN ang VIRUS na ito ay hindi maisasakatuparan... Signing off,' 'MSU Philippines' and 'BY: Someone of MSU Main Campus, Marawi City'. The payload hangs the system and shows a message accompanied by a text-mode animation of an exercising person.
Marawi.2719 BA00 01B4 402E 8B1E C901 B99F 0AE8 B401 E924 003D 9F0A 751F
- Predator.1879** **CER:** A stealth, appending, 1879-byte virus containing the plain-text 'MZCOMMANDEXECOM' and the encrypted text 'The Predator v2.44'. The payload performs a system reset. Infected files have the seconds field of their time-stamps set to 62 seconds.
Predator.1879 0E0E 1F07 B888 FFCD 213D B822 7503 EB20 9007 1F5D 5F5E 5A59
- PS-MPC.326** **CN:** An appending, 326-byte, direct infector containing the text '[PS/G²] Ender Gandalf *.COM ..'.
PS-MPC.326 8896 B602 8D96 0301 B440 B946 01CD 2133 C9B8 0042 33D2 CD21
- PS-MPC.490** **CEN:** An appending, 490-byte, direct infector containing the texts '[PS/G²] White Shark Duck Virus', '*.EXE' and '*.COM'.
PS-MPC.490 8D96 0301 B440 B9EA 01CC 33D2 B800 4233 C9CC 8D96 5A03 B440
- PS-MPC.571** **CEN:** An encrypted, appending, 571-byte, fast, direct infector containing the texts 'Written by', 'Speedo', 'Dark Angel', '*.EXE' and '*.COM'. The virus avoids infecting files with names matching the pattern '????AN*.*' (e.g. TBASCAN.EXE, TBSCANX.EXE).
PS-MPC.571 BB3C 01B9 1601 2E81 37?? ??83 EBFE E2F6
- Raymond.994** **CEN:** An encrypted, overwriting, 994-byte, direct infector containing the text 'By RiKkY moUsE Presenting the St00pid Raymond Lau Virus V1.0DUHH ==>Mocking Raymon Lau to all<== >Mock mOck moCk mocK< Quoting a chat between Raymonnd and Anon RL-> There is no such thing accomputer as a computer virus! Anon-> I think that 112MB of my hard drive might disagree with you there RL-> Oh yes like I believe you wrote a 112MB virus! And 2 or 3 things that won RL the prestigious FIDO IDIOT award! How did you know Im still a Virgin? Will you be my friend? Everybdy knows that you dont put healthy computer next to one sick with computer virus and make healthy computer sick too! Please guide me through this darkness to see the light even though I wont listen to you?'
Raymond.994 B9E2 03BA 0001 B430 80C4 10CD 21FE 06E2 04E8 0100 C3BB 5E01
- RedScorpion.1965** **CER:** An encrypted, appending, 1965-byte virus containing the texts 'Red Scorpion Copyright (C) 1995-1996 Teak Software.' and 'Ver Red.1912'. The virus avoids infecting files with names containing one of the following strings 'CO', 'IO', 'OS', 'WI', 'SC' and 'V'. Infected files have their time-stamps set to 58 seconds and n*8+3 minutes (3, 11, 19, etc).
RedScorpion.1965 05C3 B8?? ??E8 4307 B933 072E 8A14 86E0 32D0 2E88 1446 E2F3
- Rubix** **CN:** Two variants of an encrypted, overwriting virus containing the text 'Well, this is a new overwriting virus. K00l, huh? Not really. But it encrypts different sections of itself during executioner, and that's neat. Coder: Executioner'. The one-byte difference is caused by one extra NOP instruction.
Rubix.421 1600 BE2D 01BF A801 E801 00C3 608B C630 0446 3BF7 75F9 61C3
Rubix.422 1600 BE2D 01BF A901 E801 00C3 608B C630 0446 3BF7 75F9 61C3
- ShyDemon** **CN:** Two variants of an encrypted, appending, direct infector, containing the texts '(C) Shy Demon Is A Dark Wizard 1996 Production Hello... Hope I'm Not Disturbing.... Don't Wanna Cause Any Trouble... But I Just Infected 2 Files... And If You're Not Nive To Me I Will Infect More... Please Don't Kill Me! We Virii's Also Have A Life You Know... See Ya Next [03.30], Gotta Run...', '[Shy Demon - Dark Wizard - Sweden - 96.03.10]', '*.com', 'edit.com' and '\DOS'. The virus changes its decryption routine, swapping between two hard-coded variants. Infected files have the byte CCh at offset 0005h. Due to their duomorphic nature, two patterns are required for each variant.
ShyDemon.1603 2E8B 9644 07B9 E602 908D 9E37 012E 3117 4343 E2F9 5A5B 59C3
ShyDemon.1603 2E8B 8644 078D 9E37 01B9 E602 902E 3107 4343 E2F9 5859 5BC3
ShyDemon.1608 2E8B 8649 078D 9E38 01B9 E802 902E 3107 4343 E2F9 5859 5BC3
ShyDemon.1608 2E8B 9649 07B9 E802 908D 9E38 012E 3117 4343 E2F9 5A5B 59C3

VIRUS ANALYSIS 1

Zohra the Geek?

Snorre Fagerland
University of Bergen, Norway

Some months ago, the Spanish virus writer calling himself Wintermute published Zohra.4488. He claimed it contained a uuencode/uudecode routine. While they do exist in a few viruses, these routines are almost exclusively used to enable those viruses which contain binary executable code to survive transportation in text format (a good example is the BAT.Blah series). I saw no point in using uu-coding in a DOS file virus, so decided to pick Zohra apart.

Decryption

Zohra's decryption routine consists of a double decryptor. The outer loop is a polymorphic routine with standard word-sized XOR decryption. The inner is a constant decryptor doing a more complex byte-sized operation involving NOT, AND, OR, ROR and ROL. This could be what the author refers to as 'uudecode' – but it is not. The encrypted bytes are not converted into the 64-byte ASCII spectrum that constitutes the uuencode set.

Installation and DOS Emulation

On entry, Zohra makes its 'Are you there?' call – Int 21h with AX=DB15h and SI=029A. If the virus is active, SI=29A0h is returned. If not, it checks the DOS interrupt vector and, if the vector segment is higher than 0300h, emulates its way into the DOS handler.

The emulator is not very advanced; basically, it follows jumps and ignores opcodes that are insignificant to program flow. The author also omitted to set up a proper stack – maybe he was too lazy, or felt a more advanced emulator would be too big. Either way, the emulator is incapable of following CALL/RET structures. Also, a number of 32-bit, 386-specific instructions are not supported. In these cases the emulator bails out, transferring execution directly to the host. Despite its limitations, the emulator works relatively well and, in most cases, will find the original DOS handler. The only place I have seen it fail is in a DOS box under *Windows 95*.

On exit from the emulator there is a trick – a trace stack-trap. This used to be an anti-virus technique, preventing viruses tunnelling down to DOS from using interrupt tracing. Requiring only a few bytes, it was very difficult to bypass when installed in memory. However, virus writers are known to steal ideas, and several viruses now employ this tactic. It is used partly as an anti-heuristic trick, partly as a nuisance to debuggers, and partly to stop monitors tracing to the original DOS handler. In this case, execution flow runs into a terminate statement, if traced.

When it has found the DOS file handler, Zohra scans the MCB chain to find the last memory block. It checks if the block is free, and if so, shrinks it by 2150h bytes. The virus moves two copies of its own code into this area – one for use as active, resident code and the other to be encrypted and appended to host files during infection. An encryptor template is also copied to the start of this memory block. This overwrites the loader code, which serves no further purpose. Zohra then hooks Int 21h from the vector table and returns control to the host. The virus is now resident.

Stealth

Zohra uses stealth in most circumstances. It hides its size increase on DOS functions 11h/12h (FindFirst/Next) by subtracting its size from the file size in the FCB. It also masks the file time, showing the seconds field as 22 instead of 62. This directory stealth is only performed if the active process is COMMAND.COM. This trick was first used in the Dark_Avenger.2000 series, exploiting the fact that the command shell is the only program to run with the owner field in the PSP pointing to itself. Thus, Zohra's stealth mechanisms prevent tell-tale errors from utilities like CHKDSK, which will see the correct (infected) file size.

DOS functions 4Eh/4Fh are stealthed in much the same way as 11h/12h, the differences being that the masking is done in the DTA and that there is no command-shell limitation. However, Zohra determines the path-name of running processes by parsing the environment segment. Stealthing will not take place if the program making the call is F-PROT.EXE from *FRISK Software* – the virus writer was presumably aware that *F-PROT* is capable of detecting stealth activity on file size operations.

On DOS function 3Dh (Open File), the infective size is subtracted from the file's size field of the System File Table (SFT). Excluding those made by *F-PROT*, file reads are *not* stealthed. When called on infected files, the DOS function 5700h (GetTime) is designed to show the seconds field as 28, but this routine is never called. Zohra's author seems to have forgotten that he switched the high and low bytes of the function number. Thus, stealth will take place on function 0057h, not 5700h – that is, never.

A while ago I heard of something called MCB stealth, in which the PSP address of any executed program is put into the owner field of the virus' MCB. This causes memory mappers (e.g. MEM.EXE) to assume that the chunk of memory occupied by the virus belongs to the memory mapper itself, and that it should not be counted. Zohra uses a slightly different technique to the same end. When a process other than MEM.EXE is running, the virus' memory area is allocated. Meanwhile, the last free memory block has been shrunk, and the virus resides on top of the

MCB chain, out of the reach of programs needing memory. When MEM.EXE starts, Zohra adds 0212h paragraphs to the empty block directly below itself, expanding it to cover most of the virus code. Therefore, MEM only sees a large, free memory block instead of a free block and a virus. On termination of MEM, Zohra shrinks the block again, having left three paragraphs allocated when MEM started. This will show up as a total memory shrinkage of 48 bytes, which normally will not arouse suspicion.

Infection

Zohra will not infect files that include 'TB', 'AV', 'SC' or 'IV' in the path statement. This excludes many anti-virus programs – *ThunderByte*, *AVP*, *AVAST!*, *VirusScan* and *InVircible* come to mind. MEM.EXE and WIN.COM are also avoided. If the latter is executed, Zohra restores the original Int 21h handler and goes passive to avoid conflicts with *Windows*. The virus does not infect files that have the seconds field set to 62. Potential hosts must also have a creation date different from the current one – a familiar anti-goat and anti-baiting trick.

With the exception of COMMAND.COM, Zohra infects COM files between 8000 and 60793 bytes. The first three bytes of the host are replaced with a jump to the appended virus code. Candidate EXE hosts must be larger than 8000 bytes. The CS:IP, file size, stack pointer and minimum extra memory allocation fields of the EXE header are modified to reflect the file's infected state.

During infection, Zohra hooks two interrupts in addition to Int 21h. These are Int 24h (Critical Error) to suppress write protect errors on diskettes, and Int 00h (Divide by Zero), which is used as an Int 21h proxy. To execute file operations during infection, Zohra does a DIV 0 instead of calling Int 21h. This may mask the action from heuristic scanners, but it seems more like something the author threw in to be 'clever'. However, it is not clever, just convoluted, but many virus authors seem unable to make that distinction. File seek operations are performed directly in the system file table by modifying the current file pointer.

Polymorphism, Encryption and Trickery

Zohra is, as mentioned, a doubly-encrypted virus. The outer loop is a polymorphic routine with a lot of garbage instructions. The set of instructions used is relatively small as its author used a very ineffective way of generating them – hard-coding every one into decision branches. Some of the instructions are a nuisance when decrypting the virus manually; breakpoints, stack-traps and one DOS interrupt occur regularly. The inner decryption loop uses a byte-sized key, and its purpose is to decrypt the first portion of the virus. If *this* is what the virus writer calls 'uudecoding' he is wrong again, and for the same reasons.

In addition to the files Zohra will not infect, there are some other 'features' that are aimed at making things difficult for anti-virus programs. If *F-PROT* tries to read an infected file

while the virus is active, its read buffer is filled with garbage from the interrupt table. A routine directed against programs matching AV*.* sets the file size to zero in the SFT when files are opened. I have not been able to make this code run when *AVPLITE* or *AVP* opens a file, so it might be targeted against some other anti-virus product. There is also code that seems to be directed at *InVircible* (IV*), and returns carry set (call failed) on Open File or Create File operations. I have not tested this, but an educated guess is that it snubs *InVircible*, when trying to create checksum files or when opening files for checking.

Zohra has a non-destructive payload, activating on 14 April of any year. On execution of any program, it will first display some random characters and colours, then halt the computer with the message 'Zohra will live forever ! Necromancy with her ...'.

Conclusion

Zohra.4488 is a product of wasted ingenuity. Semi-accomplished virus authors often fill their creations with fancy tricks and doodads – things that are meant to show they are heavy-duty, virus-creating, assembler experts. What they do not realize is that these tricks rarely make the virus better suited for 'survival', and often work against it.

Zohra.4488

Aliases:	None.
Type:	Resident, stealth file infector.
Infection:	COM and EXE files, checks for EXE extension and the MZ/ZM marker.
Length:	4488 bytes.
Self-recognition in Memory:	Interrupt 21h, AX=DB15h, SI=029Ah returns SI=29A0h.
Self-recognition in Files:	Infected files have the seconds field of their time-stamps set to 62.
Hex Pattern in Files:	None possible.
Hex Pattern in Memory:	9D86 E03D 0057 74E2 3C11 7491 3C12 748D 3C4C 751A 2EFE 0E54
Intercepts:	Interrupt 21h: functions 0057h (bug), 11h, 12h, 3Ch, 3Dh, 3Fh, 4Bh, 4Ch, 4Eh, 4Fh, and DB15h. Int 24h during infection (disabled). Int 00h during infection (rerouted Int 21h).
Payload:	Displays a screen effect and message on 14 April.
Removal:	Boot from a clean system disk; delete and replace infected files.

VIRUS ANALYSIS 2

What a Mess!

Jakub Kaminski
Cybec Pty Ltd

From the moment the first Concept virus appeared on the scene, the evolutionary development of code and ideas implemented in macro viruses has been unavoidable. The broad spectrum of these viruses found in the wild reflects not only the current 'state of the art', but also new opportunities and potential trends. On the one hand, there is the one-macro Minimal virus (with some variants as short as 90 bytes), and on the other, viruses like Friendly, Xenixos and various virus creation kits with tens of different macros copied into infected documents.

With eighteen different macros and almost thirty in infected templates, the Mess virus is a new addition to this group of giants. Fortunately, it's a gentle giant.

Macros

There are 21 macros in documents infected with Mess and 29 in infected global templates. Those present in infected documents are: AutoExec, AutoOpen, BEEPER, CROM, CUS, CUST, EOP, ESA, ESAA, FileOpen, FileSave, FileSaveAs, INFO, MESSA, NIZ, PLT, POO, README, TheVWarning, WATCH and WEV. Infected global templates have all these (with POO renamed to OOP) plus FileTemplates, OEX, OOP1, Organizer, ToolsCustomize, ToolsCustomizeToolbar, ToolsMacro and ViewToolbars.

After taking a closer look at both lists and the macro code, it becomes clear that, in reality, there are only eighteen different macros – AutoExec (OEX), AutoOpen (OOP1), BEEPER, FileOpen (EOP), FileSave (ESA), FileSaveAs (ESAA), FileTemplates (PLT), INFO, MESSA, Organizer (NIZ), POO (OOP), README, TheVWarning, ToolsMacro (CROM), ToolsCustomize (CUS), ToolsCustomizeToolbar (CUST), ViewToolbars (WEV) and WATCH. The macros can be divided into three groups based on their function: replication, stealth and payload.

Macros containing code that copies the necessary macros into a new document or template are: AutoExec (OEX), AutoOpen (OOP1), FileOpen (EOP), FileSave (ESA) and FileSaveAs (ESAA). Those containing code that either hides the virus' presence, or makes disinfection more difficult, are: FileTemplates (PLT), Organizer (NIZ), ToolsCustomize (CUS), ToolsCustomizeToolbar (CUST), ToolsMacro (CROM), ViewToolbars (WEV) and WATCH. Code to display messages, intercept some keystroke combinations, and set or raise alarms is contained in the macros BEEPER, INFO, MESSA, POO (OOP), README and TheVWarning.

Opening an Infected Document

On opening an infected document, *Word* passes control to the virus' AutoOpen macro. After disabling auto-macros and making sure that its execution will not be interrupted, the virus assigns the short-cut key 'Ctrl-Alt-Shift-K, K' to its TheVWarning macro. Mess then checks the STARTUP-PATH directory for the file WRDBASIC.DOT. If this file is found, execution continues with the virus' replication phase; otherwise it searches the USER-DOT-PATH directory for the file THEVWARN.ING (if it exists). If found, this file is deleted.

Next, Mess collects information on the template files in the STARTUP directory, then copies the current document there under the name THEVWARN.ING. It reads the current date, stores it for further use, then changes the system date. The new date depends on the version of *Word*. Under *Word* 8, the date is set to 26 July 1995; otherwise it uses 31 March 1994. Although the virus works in *Word* 6, if it is in use, a 'Command failed' message box is displayed.

After renaming the POO macro to OOP in the previously-created THEVWARN.ING document, Mess closes the active document (after saving it) and immediately reopens it. The only reason for this is to change the document's date-stamp. Using the list of templates collected earlier, Mess unloads all global templates, sets their attributes to 'read only' and tries to delete them (the latter does not work, for obvious reasons). Mess again saves the active document as THEVWARN.ING – this time to the TEMPLATE directory – and renames its POO macro to OOP. The file THEVWARN.ING in the STARTUP directory is then renamed WRDBASIC.DOT and loaded into *Word* as an additional global template. As a result, attempts to disinfect the virus manually by deleting the infected NORMAL.DOT and letting *Word* create a new, 'clean' one will see the system reinfected by this template in the STARTUP directory.

Mess installs two items in the Help menu – 'About The 'V' Warning' and 'The 'V' Warning System Info' (assigned to macros README and INFO respectively). It copies 29 macros to the template on which the document is based (usually NORMAL.DOT) and depending on the current time, sets an alarm to 4:30, 12:10, 15:30, 18:20 (to trigger the BEEPER macro) or 23:59:59 (to trigger the MESSA macro). Other macros used in the virus replication mechanism have similar functionality and can be described as subsets of all the above features.

Stealth

Mess uses macros to perform several stealth functions – to hide its presence, to intercept calls to system macros (like ToolsMacro or FileTemplates), to skip executing standard

code and simply to return to the main program. As a result, the menu items that usually help identify the list of currently active macros are useless if Mess is active.

The WATCH macro, however, has a slightly different function. It acts as a watchdog, making sure that nothing or no-one overwrites its crucial macros in the global template (or at least, it ensures they are not 'missing' for long). In cooperation with the OOP macro, and on a regular basis (once every minute!), it refreshes some of its macros (e.g. AutoExec, AutoOpen, Organizer, ViewToolbars, etc).

Payloads

Mess has several simple payloads. Fortunately, none of these are malicious and those who do not mind having an extra twenty-something macros in their documents may even enjoy some of the 'bonuses'.

Mess' payloads include inserting text into documents and displaying message boxes. The message boxes of the BEEPER and README macros are reproduced in Figures 1 and 2, below. The INFO macro displays a message



Fig 1: The BEEPER macro produces a series of 180 beeps and displays a message box similar to this.

box containing operating system and hardware details such as version numbers and CPU type, while the MESSA macro produces one hundred beeps and then inserts the text 'THE 'V' WARNING MESSAGE Sorry to interrupt You. I think You are tired, because You have worked until mid-night. so I suggest You go to bed now and tomorrow You could work harder than this day. Kota Pelajar, Yogyakarta' in red and blue at the beginning of the current document.

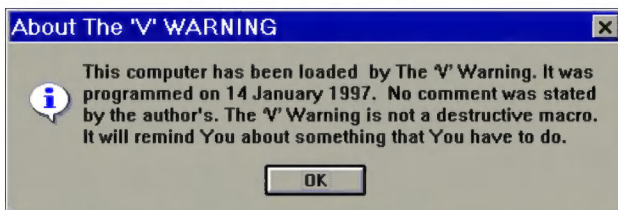


Fig 2: The README macro displays this message box.

Tidying up the Mess

Mess is unusual in yet another way; it can disinfect itself. The TheVWarning macro deletes all the other macros and menu items added by Mess, thus removing everything except itself and its short-cut key. Since the macro can only be triggered by the sequence of keystrokes described earlier, once activated, it rewards the user with a series of beeps and a message box.

What's in a Name?

The Mess virus is quite 'schizophrenic', as if it cannot decide what to concentrate on. It implements a few stealth tricks (by intercepting macros which could be used to identify viral macros) and tries hard to spread (by placing an infected template in the STARTUP directory and regularly refreshing its crucial macros) but at the same time it waves its hand in the air shouting 'Look, I'm here!' (installing new menu items, producing sound effects and displaying a number of messages). Quite a mess!

Mess	
Aliases:	None known.
Type:	Word 6/7 macro infector.
Trigger:	Various, time-based - see text. Keystroke sequence Ctrl-Alt-Shift-K, K.
Payload:	Various message boxes and text insertions. Uninstalls viral macros.
Disinfection:	In a clean Word environment, delete all virus macros through Organizer.

ADDENDUM

Memorial Revisited

In his analysis of Win95.Memorial (VB, September 1997, p.6), Péter Ször noted:

'... Next, it reads the last five bytes of the program and checks whether they start with 'SN'. If so, the file is not infected. I can see no reason for this other than the inoculation of a certain PC - the virus writer's own?'

Peter recently informed us that, on reviewing his analysis, he realized that Memorial does not check for 'SN', but for 'NS'. He then checked all the 'standard' Windows 95 COM files and found that CHOICE.COM, DISKCOPY.COM, DOSKEY.COM, FORMAT.COM, KEYB.COM, MODE.COM, MORE.COM and SYS.COM all have the string 'ENUNS' at the end (i.e. all of the Windows 95 COMs except EDIT.COM which is, internally, an EXE). After this string there is a checksum word which holds a partial checksum of the file.

The reason for this, and why Memorial avoid such COM files is that they have an internal self-check. When this routine detects a change, the host program just terminates without even displaying an error message. Regardless, it is fairly obvious that something has changed, so Memorial simply avoids infecting such 'protected' programs to reduce the likelihood of detection.

VIRUS ANALYSIS 3

Fighting Talk

Igor Daniloff
DialogueScience

A while ago I was emailed a number of complex polymorphic viruses by a mathematics student at Moscow State University. The family was later named RDA.Fighter, after the virus writer's own phrase 'Random Decoding Algorithm'. Although these viruses were zoo specimens, the 7408-byte variant has been distributed via a pirated copy of *Dr. Web v2.12* on a BBS. In this way, the virus writer introduced his 'masterpiece' to the world. Leaving this point to his conscience, I will only say that the RDA.Fighters are indeed interesting to the specialists. They deploy new and unconventional techniques, showing the author's prime objective of making these viruses almost undetectable and thus incurable.

RDA.Fighter.5871, 5969, 7802, and 7868 are advanced, TSR, polymorphic parasites designed to infect COM and EXE files. RDA.Fighter.7408 is a multi-partite virus that infects COM and EXE files and hard disk MBRs. Along with versions 7802 and 7868, it hides the size increment of infected files. In addition to the random decoding algorithm, RDA.Fighter tends to use algorithms drawn from other viruses. For example, the polymorphic engine (APE) is taken from the Phantom1 virus, the self-restoring Hamming code from the Doodle family, and the MBR infection mechanism (for RDA.Fighter.7408) from both SVC and One_Half.

Initial Decryption – APE

The outer layers of decryptors are more or less standard polymorphic decryptors, except for a few peculiarities. The decryptor is full of junk and register-twiddling commands. In most polymorphic viruses, such commands are often pure garbage – not so in this case, as the decryptor actually depends on the result of the operations.

The decryptor also contains Int 21h calls to 'non-critical' functions (values returned in the AX register do not interfere with the decoding process). Functions used include AH=30h (get DOS version), AH=35h (get interrupt vector), AH=48h (allocate memory) and many others.

Special mention must be made of the polymorphism in the RDA.Fighter.7408, 7802, and 7868 variants, because they use up to sixteen levels of decryptors which decode each other in turn. This makes detection rather tedious and time-consuming. Another factor which adds to the annoyance is that the decryptors are often backwards-decrypting. That is, you cannot set a breakpoint and are forced to single-step or run through the decryptor, unless you have one of the more powerful debuggers such as *Soft-Ice*.

Secondary Decryption – RDA

The secondary decryption process is one of the most interesting features with these viruses, which I will describe in some detail. Following their initial decryption, RDA.Fighter.5871, 5969, and 7408 install their own Int 01h handlers for single-step execution of instructions. After the virus sets the trap (trace) flag, the virus' Int 01h handler is called for every instruction executed.

Then follow some nasty instructions, which will never be executed directly – in fact, they would probably cause a system lockup if they were. These instructions can be run and debugged only when the virus' Int 01h handler is active in memory. Needless to say, this makes debugging with a standard debugger difficult. It is particularly hazardous to debug RDA.Fighter.5969 and 7408 because in subsequent activities they continue switching the trap flag off and on for dynamic correction of their own CRCs.

These two variants also contain some code in the decryptor that overwrites two sectors – one in each of two randomly-selected cylinders of head zero. This code will be run if there is an attempt to set a breakpoint in the code – the Int 01h lookahead will see the CCh and direct program flow to the payload. In RDA.Fighter.5871, the equivalent code is slightly different, and due to a bug it will never run when the trap flag is set.

When Int 01h is called (on each instruction when the trap flag is on), the virus' handler deletes the stack contents, then checks the computer type from the byte at the BIOS address FFFF:000E. If this value is FCh, the computer is an AT, and the virus takes the real time from the CMOS buffer. If not, it takes the time from the system clock via Int 21h AH=2Ch. This is the main time reference used to create a random 16-bit number for decrypting the next fragment of code. The first 'nicety' of these viruses is hidden in this decoding routine.

After a lot of multiplications, divisions, and subtractions of a 32-bit random number, a 16-bit random number is left in the AX register. Masking this number with 1Eh gives a random pointer into a table containing instructions to be used in the decryption. There are two tables, each containing sixteen decrypting commands, of which a few are repeated. We will call them the primary and mirror tables, and they are presented below.

Decryption at this stage takes place in 16-iteration cycles. On the basis of the values stored in AX, a decrypting command is chosen from the primary table depending on the DI register contents. For example, if a bit is set at one particular bit position in AX, then the command residing in that position in the table is copied from the primary table to the decryptor. An initial constant, which in a decrypting

Primary table	Mirror table
XOR [BX], DX	NOT [BX]
ROR [BX], CL	NEG [BX]
ROL [BX], CL	SUB [BX], DX
SUB [BX], DX	ADD [BX], DX
ADD [BX], DX	XOR [BX], DX
NEG [BX]	ROR [BX], 1
NOT [BX]	ROL [BX], 1
INC [BX]	INC [BX]
DEC [BX]	DEC [BX]
ROR [BX], 1	NOT [BX]
ROL [BX], 1	NEG [BX]
XOR [BX], DX	SUB [BX], DX
SUB [BX], DX	ADD [BX], DX
ADD [BX], DX	ROR [BX], CL
NEG [BX]	ROL [BX], CL
NOT [BX]	XOR [BX], DX

cycle is dynamically varied according to a given law, is also copied there. This means that on each decrypt cycle AX will vary and so will the structure of the decryptor.

On termination of a decrypting cycle, the value of DI is set to the next command in the primary table. This operation is then carried out for all set bits in the 16-bit number. If no bit is set in any position, the decrypting cycle is skipped and the value of DI is increased to the next command. If the DI pointer overshoots the last command, its value is set to the first command in the table.

When this operation is completed, the virus begins to check whether all smart combinations have been deciphered, in an equally smart way. It computes the decrypted code fragment's CRC as a function of the fragment that was not encrypted. The CRC computation pattern is 'fan-shaped' from the decryption boundary and upwards, and a 48-bit number is computed downwards. CRCs are computed using all possible transfer commands ADD, XOR, etc.

RDA.Fighter.5969 and 7408 dynamically apply additional CRC corrections via Int 01h. RDA.Fighter.7408 has four further forms of CRC check, which vary from copy to copy. The computed CRC is stored in the AX, BP, and DX registers. BP is summed with the number stored in the virus body and corrected for finding the base point of the virus in memory. The offset of BP is compared with the value of DX. For correct decoding, the CRC storage offset must lie in the previously-encrypted fragment.

If the value of the CRC does not match that of DX, the decryption attempt is 'undone' using the complementary commands from the mirror table, thus reversing the procedure. This resets the incorrectly-decrypted fragment so the virus can repeat its attempt at 'correct' decryption, starting with the determination of a random number in AX using the primary table commands.

The last two modifications of RDA.Fighter.7802 and 7868 apply the same random decoding algorithm as before, but with vital changes. The virus writer named this improved mechanism the Random Decoding Algorithm Engine (RDAE). After a maximum of sixteen APE-generators, these viruses have a polymorphic random decoding

mechanism. In other words, the commands that implement random decoding alternate with garbage commands which in no way affect the decoding process. All jumps, references, and offsets used in RDA decoding are corrected by the virus while this code is being generated. The primary table and its corresponding mirror table will be different in files infected by the 7802 and 7868 variants.

Detection of RDA.Fighter.7802 and 7868 is indeed a Herculean task. It is even more difficult to cure infected files, because full and correct decoding with a random key depends on the code implementation of the actual RDA decryptor created for that infection.

These viruses deciphered their own code within seven seconds on a test 386DX-33 MHz machine using 'RandomDecodingAlgoritm' (note the author's typo). Surprisingly, on a Pentium 100 MHz machine, the tests took longer. I found the main determinant of decryption speed was not the random sequence of base values used in the encryptor, but a pseudo-random factor more dependent on processor speed (where randomness decreases as speed increases). I have RDA.Fighter replicants that refuse to decipher themselves on some processor types. My company's scanner, *Dr. Web*, does not use the system clock for determining random base values and thus deciphers the RDA viruses in 40-400 test cycles.

Installation

Once fully decrypted, the virus checks the word at 0000:00C5h (the Int 31h handler address in the Interrupt Table). If it contains the characters 'SF', and the byte at 0000:00C7h corresponds with the virus' version number (10h for RDA.Fighter.5871, 0Bh for 5969, 14h for 7408, 1Fh for 7802, and 20h for 7868), the virus assumes that it is already resident, and hands control to the host application.

If not, the virus traces Int 13h to its original handler. The Int 21h handler is found by searching the DOS segment for 9090E8h (NOP; NOP; CALL) or FA80FC6Ch (CLI, CMP AH, 6Ch) – code commonly found at the start of Int 21h.

When the virus finds the Int 13h original handler, it reads the second sector of the first hard disk into memory, writes the word 'SF' at the buffer start, and writes this buffer back to the second sector via the original Int 13h. Finally, this sector is again read via Int 13h, compared with what should have been written, the original is restored and the sector written back to the disk in its original form. If the comparison reveals a mismatch, the virus assumes that the disk is cached and does not intercept Int 13h during infection.

To go resident, the virus reduces the memory allotted to the infected program (Int 21h AH=4Ah), intercepts Int 21h, starts the infected program (Int 21h AX=4B00h), and removes itself from memory (Int 21h AH=49h). It gets the return code (Int 21h AH=4Dh), fetches the memory block it had taken from DOS, and completes the process (Int 21h AX=4C00) by installing its resident copy in memory.

Self-preservation and Stealth

The resident copy applies a powerful, noise-proof algorithm (Hamming code) for restoring its code, making debug breakpoints useless. Furthermore, as a 'trick' to mislead the researcher, the virus introduces an instruction for exiting from interrupt (IRET) in its Int 21h handlers. However, RDA.Fighter corrects the handler using its noise-proof restoration algorithm. In testing, the virus restored sixteen and more sequentially changed bytes in the virus body!

RDA.Fighter.7408, 7802, and 7868 control the find file functions (Int 21h AH=11h, 12h, 4Eh, 4Fh) to hide the size increment of infected files.

Infection

The resident part of the virus controls the DOS File Open (AH=3Dh) and Load/Execute program (AX=4B00h) functions. RDA.Fighter.5969 controls the file rename function (AH=56h) as well. When these functions are called, RDA.Fighter tries to infect the files indicated by these functions.

RDA.Fighter infects COM and EXE files of 4096 bytes or larger, but avoids files matching *ES?.* (AIDSTEST.EXE), *WE?.* (WEB.EXE), or *AN?.* (COMMAND.COM). Files with the system attribute are not infected, and a prospect's file format is checked by the file extension and the 'MZ' signature of EXE files. While infecting files, the virus determines the drive letter from the filename (Int 2Fh AX=121Ah) and the free space on the disk (Int 21h AH=36h). The virus uses the seconds value in the host's time-stamp as an infection flag (RDA.Fighter.5871 – 32 seconds; 5969 – 22; 7408 – 40; and 5969 and 7868 – 2).

During infection, the virus installs its own Critical Error handler (Int 24h), and Disk I/O handler (Int 13h). If there is no disk cache program, it also installs handlers for Int 01h (single step execution of commands), Int 03h (breakpoint) and Int 2Ah (*Microsoft* Network) – these handlers are all a single IRET (Return from Interrupt).

Initially, the virus code occupies about 600 bytes. In infected files, along with the garbage commands, it takes up about 1500 to 4000 bytes.

It should also be noted that the virus encrypts a piece of the host program at a random offset. When loading an infected program, the virus decodes and restores this encrypted fragment. The offset and code fragment to encrypt is randomly chosen, in much the same way as it determines the random AX in the decryptor. This procedure was probably designed to prevent anti-virus tools employing generic disinfection methods (AdinfExt, TbClean and others) from restoring infected files.

RDA.Fighter.7408 also infects the MBR of hard disks. Its infection algorithm closely resembles that of the viruses SVC.4641, 4644, and 4677. RDA.Fighter.7408 replaces the first three bytes of the boot-loader in the MBR with JUMP

NEAR PTR and writes 33h virus bytes in the zero-byte region. The 'tail' of the virus is written in fifteen sectors, beginning from the second sector of track zero, head zero.

The different variants contain various internal texts, and the following are occasionally displayed:

RDA.Fighter.7408:

Stealth Fighter 2.0 : New Aggression.

RDA.Fighter.7802:

Stealth Fighter DEMO Part (3.1): Enemy Unknown.

RDA.Fighter.7868:

Stealth Fighter, DEMO Part (3.2): Next mutation 06/09/95.

Analysing RDA.Fighter was quite a challenge, but I succeeded in adding full detection and disinfection to *Dr. Web*.

RDA.Fighter

Aliases:	APE.RDA.
Variants:	RDA.Fighter.5871, 5969, 7408, 7802 and 7868.
Type:	Memory-resident, polymorphic file infector. RDA.Fighter.7408 is multi-partite.
Infection:	COM and EXE files. RDA.Fighter.7408 MBR also.
Self-recognition in Files:	Seconds field of file time-stamp. See text for details.
Self-recognition in Memory:	The string 'SF' at address 0000:00C5.
Hex Pattern in Files:	None possible.
Hex Pattern in Memory:	<p>RDA.Fighter.5871:</p> <pre>50FE C43D 004C 740B 80FC 3E74 0658 2EFF 2EEF 17E8 E50F 2EC7</pre> <p>RDA.Fighter.5969:</p> <pre>50FE C43D 004C 7410 80FC 3E74 0B80 FC57 7406 582E FF2E 5118</pre> <p>RDA.Fighter.7408:</p> <pre>50FE C43D EEEF 7505 EBDE 0158 CF3D 004C 745B 80FC 3E74 5680</pre> <p>RDA.Fighter.7802:</p> <pre>50FE C43D EEEF 7505 E88B 0158 CF3D 004C 745B 80FC 3E74 5680</pre> <p>RDA.Fighter.7868:</p> <pre>50FE C43D EEEF 7505 E88E 0158 CF3D 004C 745B 80FC 3E74 5680</pre>
Intercepts:	Int 21h for infection.
Payload:	Displays messages (see text for details).
Removal:	Using a clean system, identify and re-store infected files.

PRODUCT REVIEW

Inoculan AntiVirus v5.0 for Windows 95

Dr Keith Jackson

Computer Associates claims its product is 'a full-featured *Windows 95* application that detects and removes viruses'. In other words a scanner, and both on-demand and memory-resident components are provided. *Inoculan's* packaging claims '100% protection, 100% cure against all macro viruses', and 'Automatic Protection Against Virus Attack – GUARANTEED'. The latter claim is already dead in the water – nothing provides guaranteed protection against virus attacks. Anyone who claims that their product does is either lying or does not understand the problem.

Installation

Inoculan was provided for review on a single CD-ROM. Well, actually, two consecutive CD-ROMs were provided, as the first one refused to install. Installation always appeared to finish correctly, but the error 'A required .DLL file, WININET.DLL, was not found' popped up whenever I tried to execute *Inoculan*. As I review anti-virus software on a PC that is not networked, it seemed an unlikely error. The problem persisted whether I accepted the proffered defaults, or ran a custom installation. Changing the setup of *Windows 95* made no difference. Several infuriating attempts later, this error was fed back to *Inoculan's* developers, who agreed that it did produce the stated error. Hmmm, so much for quality control.

Using another CD containing a new build of *Inoculan*, installation proved to be very straightforward, if a little odd. Instead of the usual SETUP program, this CD had one file called CAV50B47.EXE. Given the lack of alternatives on offer, I executed it. A message box appeared saying 'This will install *Inoculan AntiVirus 5.0*, do you wish to continue?' I said 'Yes', and the InstallShield program extracted the relevant files while SETUP appeared as in my previous, abortive attempts. There is a licence screen to read, name and company details to be entered, and a choice to be made between Typical, Compact and Custom installation. 'Compact' is barely explained, so I chose the 'Typical' option. I selected where *Inoculan's* files were to be stored, the mandatory bar-graphs whizzed up and down, and after a reboot, installation was complete.

Web and Email Support

Now is probably the best time to mention that *Inoculan* has extensive email and World Wide Web support facilities. I browsed the support site at www.cheyenne.com to investigate my installation problems. I have seen Web sites that are difficult to follow, usually because they are providing

too much information for the minds of mere mortals. The *Inoculan* support site wins the prize for the most complicated. There are support pages for umpteen software packages, options galore, and places where you can email everybody under the sun at the developer's. Or so it seems.

I sent an email reporting my WININET.DLL problem. It whizzed off into the ether, but the next time I checked my email, it had bounced. The error message said 'User not listed in public Name and Address book'. But I was using the address generated automatically by the developer's Web site, not one that I had, perhaps erroneously, typed in. I tried again, choosing another of the myriad email addresses, and it bounced again. At this point I gave up.

Documentation

The review copy's documentation comprised a small, 60-page book. It contains a reasonable account of what computer viruses are, hints on preventing infection, and very clear instructions on installation. It is a shame that the latter did not match what actually happened with the review CDs. The advice on using the scanner, selecting various options, and scheduling automatic scans are well-written and easy to understand. Given the current trend towards not providing any printed documentation at all, *Inoculan's* book looks quite good.

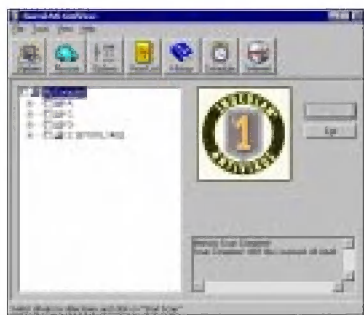
The on-line Help files are very pretty. They are adequate on routine things but the blurb provided is very superficial. There is no mention of what to do if things go wrong – probably the most important thing to be documented.

Operation

The first time *Inoculan* was executed, it moaned that a file called INFO.SIG was missing, presumably because it had to generate this file for itself. I have no idea why it would do this – no information about it is provided. Perhaps the scanner maintains a list of successfully scanned files, so that the scanning process can be sped up in future?

At this point a screen appeared offering 'Congratulations'. I am installing some routine utility software, and here I am being encouraged to keep going like a five-year old. A screen offering to register *Inoculan* via the Internet is also displayed. Now, that is hi-tech. It is also impossible on a PC which does not have a modem attached. Like my test machine. There was a huge registration sheet, with boxes everywhere, insisting that 'All [required fields] must be entered' before I could proceed. Oh God, a questionnaire.

When the *Inoculan* standalone scanner actually executes, it is fairly standard fare, with drop-down menus, an onscreen selection of what is actually to be scanned, a big list of known viruses, and onscreen buttons for the most important



functions. When a scan is in progress, a silly animation rotates a radar beam round in a circle, which seems pretty pointless.

The default scanner settings are to inspect all files, look inside a huge list of compressed

file types, and perform what *Computer Associates* calls a 'Quick Scan'. The other types available are 'Thorough' and 'Reviewer' scans. What an honour! Note that by default all files are scanned.

Inoculan can be configured so that one of several actions is taken when a virus is found. The options are to 'clean' the infected file, rename it, move it to the quarantine area, log the occurrence, or delete it. I tested *Inoculan* by merely logging the infections, and as usual would strongly recommend that the 'clean' option is left well alone.

Scanning

I tested *Inoculan*'s detection capabilities against the *VB* test-sets. Unfortunately, scanning this CD-ROM in its entirety proved too much, and at 3237 infections an error message appeared: 'This program has performed an illegal operation and will be shut down'. Not an auspicious start. I resorted to scanning the *VB* test-set section by section.

Inoculan detected all bar one of the 549 samples contained in the 'In the Wild File' test-set (it missed one of the two *No_Frills_Dudley* samples). This result was the same with a 'Quick' or a 'Thorough' scan. In fact, throughout my testing, these two types of scan always produced exactly the same detection rate. Activating the 'Reviewer' scan resulted in all of the 'In the Wild File' samples being detected. Overall, this result is as close to 100% perfection as makes no difference.

Inoculan detected only 768 of the 774 samples of the 'Standard' test-set (99.2%). It missed all three samples of *Maresme*, both samples of *Cruncher*, and one of the three samples of the *Greets.3000* virus. When 'Reviewer' scanning was invoked, the samples of *Cruncher* were both found, leaving just four samples undetected (99.4%).

Things got distinctly worse when the 'Macro' test-set was scanned. *Inoculan* detected only 682 of the 716 test samples (95.2%) – a performance augmented by just three more detections under 'Reviewer' scanning. This result is worse than that achieved by other comparable products, probably reflecting the novelty of many macro viruses and the difficulty developers are having in playing catch up.

All 91 of the 'In the Wild Boot' viruses were detected. Unfortunately, after reporting an infection, *Inoculan* generated a fatal exception, causing *Windows 95* to close it

down. This occurred regardless of which of the offered 'Clean', 'Delete' or 'Log Only' actions was chosen. That these options were presented at all was somewhat odd, as 'Log Only' was set in the main Options dialog before starting this test and the option to increase security on virus detection was disabled. After about 25 lock-up and restart cycles, *Inoculan* would falsely discover a virus in memory while loading. The only solution that allowed testing to continue was a system restart.

The 'Polymorphic' test-set currently contains 13,000 viruses (500 samples of 26 viruses), and no matter what type of scan was invoked, *Inoculan* always detected 12,483 test samples as infected (96.0%). All 500 *Cryptor* samples were missed, as were sixteen other samples. This is a good polymorphic detection rate. Adding detection of *Cryptor* would take *Inoculan*'s polymorphic detection rate very close indeed to 100%. The only problem was that after each 'Polymorphic' scan, *Inoculan* locked up solid, requiring a Ctrl-Alt-Del reboot to regain control of the PC.

Overall, *Inoculan*'s detection rate is commendable. The commoner viruses are detected with confidence, and some more work on macro virus detection (which, given its immediacy, will surely already be in place), and polymorphic detection, will take the overall detection rate close to 100%. A good effort overall.

Things were not so good when I tested *Inoculan* against the *VB* 'Clean' test-set. This comprises 5500 executable files, held on CD-ROM, all of which have been copied from well-known software products, and none of which are infected. *Inoculan* declared that sixteen of them were (ten with *Sutemi*, and six with *Number1a*). Under 'Reviewer' mode, 34 files were thought to be infected; the same sixteen, plus eighteen others with a further four different viruses. One false positive is a bad result – false positives being detected on this scale are a thorough nuisance. Much more work is needed here before *Inoculan* can be trusted not to declare 'clean' files virus-infected.

Speed

Using the fastest possible combination of scanner settings (program files and documents only, do not look inside compressed files, do not check the boot record, Quick scan), *Inoculan* scanned the C: drive of my test PC in 56.6 seconds. Just under half (666) of the 1412 files stored on the hard disk were actually scanned. Neither checking the boot sector, nor introducing the 'Thorough' scan had any material effect on this scan time. 'Reviewer' scanning more than tripled the time to 3 minutes 4 seconds. Scanning inside compressed files increased it further to 3 minutes 12 seconds (there were not many compressed files on the drive). Finally, scanning all 1412 files (with all other options still active) took 5 minutes 58 seconds.

For comparison purposes, the DOS version of *Dr. Solomon's Anti-Virus Toolkit* took 6 minutes 40 seconds, and the DOS version of *SWEEP* from *Sophos* 3 minutes 17 seconds

to perform the same scan. Both of these scanners inspected over 700 files, but even so this leaves *Inoculan* a clear speed leader.

Memory-resident Software

Inoculan's memory-resident component is called 'Real Time Monitor'. This can be configured using menu options within the product itself. However, the only ones available are: loading memory-resident software whenever the PC is rebooted; permitting memory-resident software to be deactivated at will; and scanning a floppy disk if one is found in the A: drive when *Windows 95* is shut down. That is it. It is not possible to 'tailor' the actions taken by the memory-resident software.

Inoculan's documentation states that 'All files being executed are scanned for infection before they are allowed to run'. This sounds good, but it means that infected files can be copied around at will. Sure enough, I could use the *Windows 95* 'drag and drop' features to copy entire folders full of virus-infected files without the memory-resident software issuing any complaint.

One feature of the Real Time Monitor displays a window showing the total number of files scanned, the number of files found to be infected, and the number of files 'cleaned'. I tested this by using 'drag and drop' to copy the entire 'In the Wild File' test-set of 549 samples from CD-ROM to hard disk. Initially, the total number of files scanned rose by just 42, meaning over 90% of the files had not been scanned. Now, I can understand what is going on if no files are scanned when copied, but why this small increase? When I deleted the 'In the Wild File' subdirectory from the hard disk, and tried the test three more times, the number of files scanned rose by 41, 25 and 37 respectively. I have no idea why this number is not consistent.

The Rest

Inoculan exhibited several oddities, most of which can be put down to a need for further development and/or testing. I have already mentioned the lock-up problem when lots of file viruses, or any boot viruses, are found. This is more than just a nuisance for reviewers. When a heavily-infected site is cleaned out, it is quite possible that many hundreds of files could be infected (if not thousands on a large server). Restarting *Inoculan* after every scan would prove a major distraction from the task in hand.

The 'Quarantine Viewer' allows viewing of all the suspect files in the quarantine subdirectory. The problem is that the Close button at the top right corner of this window does precisely nothing. If I can find that in just a short period of testing, why did the developers not spot it?

When a scan finds a virus and execution of the scanner is terminated and then immediately restarted, an error message always appears saying that a virus has been detected in memory. The only way to cure this is to reboot the PC.

Every single time. The only work-around I could find was to instruct the scanner not to scan memory, which removes what should be a very useful check. All this is merely a matter of the *Inoculan* scanner clearing up after it has finished, and not doing it is merely sloppy programming.

If one CD-ROM is exchanged for another, the subdirectory structure shown onscreen remains the same. It is necessary to use the 'Refresh' option from the *Inoculan* menus to force the screen to be updated. This can cause confusion, as could the disturbing habit of resetting the type of scan that is performed, without warning, if a virus is found. I do not mind the scanner doing this, indeed it has a laudable aim, but a message warning the user that this has happened would be useful.

Conclusion

Given that *Inoculan* does not come close to detecting 100% of the macro viruses in the VB 'Macro' test-set, never mind cleaning them, the claim stated at the beginning of this review falls by the wayside. It is mere marketing hype. But you knew that anyway.

Installation of *Inoculan* proved to be quite straightforward once I actually received a version that worked, but given this and the many other problems mentioned here, do insist that *Inoculan* is actually demonstrated on your PC before agreeing to any purchase.

Having said all that, *Inoculan*'s virus detection capabilities are quite good – it's the hype and stability problems that need attention, not the detection rate *per se*. *Inoculan* is also respectably quick, easy to use, and will do a capable job for most users. The things to be wary of are its unacceptably high rate of false positives and the stability problems exposed in the 'In the Wild Boot' and full test-set tests. These problems need fixing, and soon! Prospective purchasers would be well-advised to ask for evidence that these things have indeed been fixed.

Technical Details

Product: *Inoculan AntiVirus v5.0* for *Windows 95*.

Vendor: *Computer Associates International*, 1 Computer Associates Plaza, Islandia, New York 11788, USA.
Tel +1 516 465 4000, Fax +1 516 465 6600, Compuserve GO CHEYENNE, World Wide Web <http://www.cheyenne.com/>.

Availability: 486 CPU running *Windows 95*, 8 MB of hard disk space, minimum 8 MB RAM, CD-ROM drive and 3.5-inch floppy disk drive all required.

Version evaluated: 5.0 (Engine version: 4.01, 21/9/97; signature version: 4.01, 2/10/97).

Price: List price for a single user \$69.00.

Hardware used: A 133 MHz Pentium with 16 MB of RAM, a 3.5-inch floppy disk drive, a CD-ROM drive, and a 1.2 GB hard disk divided into drive C (315 MB), and drive D (965 MB). This PC can be configured to operate under either *Windows 95*, *Windows 3.11*, *Windows 3.1*, or DOS 6.22.

Viruses used for testing purposes: For a detailed listing of test-sets used for this review see VB, September 1997, p.16.

ADVISORY BOARD:

Phil Bancroft, Digital Equipment Corporation, USA
Jim Bates, Computer Forensics Ltd, UK
David M. Chess, IBM Research, USA
Phil Crewe, Times Mirror International Publishing, UK
David Ferbrache, Defence Research Agency, UK
Ray Glath, RG Software Inc, USA
Hans Gliss, Datenschutz Berater, Germany
Igor Grebert, Trend Micro Inc, USA
Ross M. Greenberg, Software Concepts Design, USA
Alex Haddox, Symantec Corporation, USA
Dr. Jan Hruska, Sophos Plc, UK
Dr. Keith Jackson, Walsham Contracts, UK
Owen Keane, Barrister, UK
John Laws, Defence Research Agency, UK
Rod Parkin, RPK Associates, UK
Roger Riordan, Cybec Pty Ltd, Australia
Martin Samociuk, Network Security Management, UK
John Sherwood, Sherwood Associates, UK
Prof. Eugene Spafford, Purdue University, USA
Roger Thompson, NCSA, USA
Dr. Peter Tippet, NCSA, USA
Joseph Wells, WildList Organization International, USA
Dr. Steve R. White, IBM Research, USA
Ken van Wyk, SAIC (Center for Information Protection), USA

No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein.

SUBSCRIPTION RATES

Subscription price for 1 year (12 issues) including first-class/airmail delivery:

UK £195, Europe £225, International £245 (US\$395)

Editorial enquiries, subscription enquiries, orders and payments:

Virus Bulletin Ltd, The Pentagon, Abingdon Science Park, Abingdon, Oxfordshire, OX14 3YP, England

Tel 01235 555139, International Tel +44 1235 555139

Fax 01235 531889, International Fax +44 1235 531889

Email: editorial@virusbtn.com

World Wide Web: <http://www.virusbtn.com/>

US subscriptions only:

June Jordan, *Virus Bulletin*, 590 Danbury Road, Ridgefield, CT 06877, USA

Tel +1 203 431 8720, Fax +1 203 431 8165



This publication has been registered with the Copyright Clearance Centre Ltd. Consent is given for copying of articles for personal or internal use, or for personal use of specific clients. The consent is given on the condition that the copier pays through the Centre the per-copy fee stated on each page.

END NOTES AND NEWS

In early November, the *Integralis* group created **two new companies from former operating divisions of *Integralis Limited*** which has since become a non-trading holding company. *Integralis Technology* is responsible for the development and marketing of *MIMESweeper*, while *Integralis Network Systems Ltd* provides network security and integration solutions. For details on all three organizations, visit the *Integralis* Web site at <http://www.integralis.com/>.

The 7th USENIX Security Symposium will be held in San Antonio, Texas from 26–29 January 1998, and will look at the latest advances in computer security, cryptography and electronic commerce. At the exhibition on 28/29 January, samples of the newest security tools and applications from 50 vendors will be available for testing. The full programme is on-line at <http://www.usenix.org/events/sec98/>, or email info@usenix.org.

From 28–30 April 1998, one of the UK's largest corporate IT functions will take place at London's Olympia. *Infosecurity* is set to join forces with *Network, Systems & Applications Management '98* and also *Customer Service & Support '98*. The event, *Network Systems & Applications Management '98*, is expected to attract over 6,000 visitors to more than 250 exhibitions. More details and contacts for all three subsidiary events can be found at the Web site <http://www.infosec.co.uk/>.

Reflex Magnetix Ltd's new disk encryption product, *Disknet Data Vault*, incorporates the Blowfish algorithm with techniques allowing transparent, on-the-fly operation. Unlocking the encrypted drive with the correct password automatically decrypts files on access. The package retails for £49 +VAT, and requires *Windows NT* and a minimum of 400 KB hard disk space to install. For further details contact Phillip Bengel; phillip.bengel@reflex-magnetix.co.uk.

Two computer virus workshops will take place next month at the *Sophos* training suite in Abingdon in the UK. An introductory course on 14 January 1998 will be followed by an advanced session the next day. A **practical NetWare Security course** is also taking place at the same site on 8 January, priced at £325 +VAT. For more details contact Karen Richardson; Tel +44 1235 544015, fax +44 1235 559935, or visit the company's Web site; <http://www.sophos.com/>.

Hi-tech public relations firm *Network Associates* has consulted its lawyers following the proposed merger of *McAfee Associates* and *Network General* into an organization of the same name, as reported in last month's *Virus Bulletin*. Is it merely irony that the public relations company represents, among other clients, an old rival of *McAfee's* – none other than *Symantec*?

Unix security and anti-virus specialists, ***CyberSoft Inc***, run **monthly Certified VFind Professional classes** at their headquarters in Conshohocken, Pennsylvania. Classes cover basic Unix security and the deployment and use of *VFind* as part of your security policy implementation. Confirmed course dates for 1998 are 22–23 January, 26–27 February and 26–27 March. The two day course costs \$850. For bookings or further information Tel +1 610 8254748 or visit <http://www.cyber.com/>.

Network Associates* has announced the acquisition of *Pretty Good Privacy Inc – makers of *PGP*, the worldwide *de facto* standard for Internet email and file encryption. *Network Associates*, formerly *McAfee Associates*, plans to continue *PGP's* progress in developing security standards with the Internet Engineering Task Force. *Network Associates' Total Virus Defense Suite* and *PGP's Business Security Suite* desktop encryption software will be amalgamated into a new product – the *Total Network Security Suite*.